容器云多维资源利用率均衡调度研究 *

龚 坤，武永卫，陈 康

(清华大学 计算机科学与技术系，北京 100084)

摘 要：基于 Docker 的容器云技术是当前研究的热点，容器云中资源的调度策略对数据中心资源利用率和集群性能具有决定性作用。在 OpenShift 容器云平台上针对其调度策略进行研究和改进，提出了基于多维资源空闲率权重的评价函数和调度方法。该方法综合考虑物理节点 CPU、内存、磁盘、网络带宽空闲率和已部署的容器应用个数等因素，并用模糊层次分析法 FAHP(fuzzy analytic hierarchy process)自动建模求解容器应用多维资源权重参数。实验表明，新的调度方案能够使集群多维资源利用率更加均衡，从而提升资源的利用率和集群性能。

关键词：容器云；调度策略；OpenShift 平台；评价函数；负载均衡

中图分类号：TP302　　doi: 10.19734/j.issn.1001-3695.2018.09.0764

Container cloud multi-dimensional resource utilization balanced scheduling

Gong Kun, Wu Yongwei, Chen Kang

(*Dept. of Computer Science & Technology*, *Tsinghua University*, *Beijing* 100084, *China*)

Abstract: Docker-based container cloud technology is the current research hotspot. The scheduling strategy of resource in container cloud is decisive for data center resource utilization and cluster performance. On the OpenShift container cloud platform, the scheduling policy is researched and improved, and the evaluation function and scheduling method based on the multi-dimensional resource idle rate weight are proposed. It considered the factors of CPU, memory, disk, bandwidth and the deployed container applications on the physical machines, used the FAHP (Fuzzy Analytic Hierarchy Process) to automatic modeling and solving the muti-dimensional reosurce weighting parameters of the container applications. Experiments show that the new scheduling enables the cluster multi-dimensional resources utilization more balanced, improves resource utilization and cluster performance.

Key words: container cloud; scheduling strategy; openshift platform; evaluation function; load balancing

0 引言

Docker 容器[1]虚拟化技术日益发展，已经成为当前容器技术的代名词。与传统的虚拟机相比，Docker 是一种共享操作系统内核的虚拟化解决方案。基于 Docker 的虚拟容器(container)和宿主机(Host)在共享系统资源的同时能实现容器间的隔离，多个容器间互不影响。应用打包到容器中可以移植到任何平台上实现快速的启动、部署和交付，容器中虚拟磁盘 IO 和 CPU 的性能甚至接近宿主机[2]。随着容器大规模集群管理技术的日渐成熟，如 Kubernetes、Mesosphere 和 Docker Swarm[3]等主流的容器管理技术发展，基于容器虚拟化技术的 PaaS(platform as a service)平台逐渐成为下一代云计算的核心。各大互联网公司正逐渐使用容器替代传统的虚拟机，出现了一大批基于容器化技术的容器云，如 Google Container Engine、OpenShift 等。

容器云中资源调度器对集群的性能和资源利用率起到决定性作用，是容器云的核心。传统虚拟机式的云计算集群对调度策略已有相当多的研究[4]，主要集中在降低系统能耗、集群服务器的负载均衡、提高数据中心资源利用率和基于经济学管理模型等几个方面。文献[5]提出一种根据虚拟机负载动态调节处理器电压来降低集群能耗；文献[6]提出来一种中心平衡器的平衡算法来均衡集群服务器的负载；文献[7]提出一种将应用需求和物理计算资源建模，减少所需物理服务器的数量来提升资源利用率；文献[8]提出了一种基于市场经济的计算资源分配策略。然而，针对容器云中容器的调度研究不多，传统的调度方式不能充分考虑容器的特点，往往造成容器云资源利用率不高和负载不均衡。容器云中资源的调度主要依靠容器编排实现，不同的编排引擎对容器的管理和调度不同，其调度方式主要有三种典型的实现方式:

a) 集中式的调度。整个集群状态存储在唯一的调度实例中，使用集中的、单一的调度算法来调度所有的作业，如 Docker 的容器编排引擎 Fig、Machine 和 Swarm[9]

b) 两层调度模型。将资源调度和作业调度分开，资源调度器只负责资源的分配，具体的作业调度由独立的作业调度框架完成，如 Mesos[10,11]和 Hadoop YARN 调度。

c) 共享状态资源调度。每一个调度器拥有对全部资源的访问权限，根据自身调度方案进行自由的竞争，实现调度器的高并发和高可扩展性,如 Brog、Kubernetes[12]等。

基于 Docker 容器虚拟化技术的 PaaS 层 OpenShift 容器云平台使用 Kubernetes 进行容器管理和调度，其资源调度方式正是共享状态资源调度模型。但是，OpenShift 容器云平台默认调度方式只考虑 CPU 和内存的因素，并未根据当前部署应用的特性和当前节点已有的应用部署数量进行调度，造成其资源利用率不高和负载不均衡。不同容器应用对资源的需

收稿日期：2018-09-20；**修回日期**：2018-11-01　**基金项目**：国家重点研发计划资助项目（2016YFB1000504）；国家自然科学基金资助项目（61433008，61373145，61572280，U1435216）；中国博士后科学基金资助项目（2018M630162）

作者简介：龚坤（1991-），男，贵州铜仁人，硕士，主要研究方向为容器云、分布式计算（gongkunjxl@163.com）；武永卫（1974-），男，教授，博导，博士，主要研究方向为并行计算；陈康（1976-），男，浙江台州人，副教授，硕导，博士，主要研究方向为分布式计算、存储系统.

求不同，OpenShift 仅考虑几个单一因素的调度方案不能实现多维资源利用率的负载均衡，容易造成容器云集群中某些节点过载，从而产生资源浪费和负载不均衡。

针对 OpenShift 默认调度方式存在的问题，本文提出了一种新的调度方案和函数评价体系。该方法将函数评价体系分为空闲率评分和反馈评分两个模块，综合考虑物理节点 CPU、内存、磁盘、网络带宽空闲率和已部署的容器应用个数等因素。通过感知用户的资源需求，利用 FAHP 模型对应用多维资源需求建模和并自动求解满足模糊判断矩阵一致性检验的多维资源权重参数。将资源权重参数应用到评价函数中，选择评分最高的节点作为调度目标节点。新调度方案能够使集群资源负载更加均衡，提升集群资源利用率。

1　OpenShift 资源调度策略

1.1　OpenShift 调度流程

OpenShift[13]是红帽公司提供 PaaS 层容器云平台，支持多种开发语言和框架，允许开发者或开发团队在此平台上快速创建、测试、部署以及运行应用，OpenShift 私有云平台对容器应用默认调度流程如 1 所示。客户端 Web Browser 和 Kubectl (Kubernetes command-line tools)首先向 API Server 发送特定的应用请求如创建 Pod、然后 API Server 将用户请求的响应处理结果存储到 Etcd 中。调度器在 API Server 响应完成后从未调度的 Pod 队列中弹出一个 Pod 对象、接下来有两个阶段的工作。第一阶段：预选节点。检查每个物理 Node 节点是否满足 Pod 的资源需求，主要包括包括端口冲突、CPU、内存容量检测、服务占用等，淘汰不满足需求的物理节点。第二阶段：优选节点。将剩余的节点根据 CPU 和内存空闲资源率进行综合的评分，按照评分的大小排序，选出最合适的节点。最后将 Pod 绑定到选出的目标 Node 节点上，调度器完成容器应用一次调度。

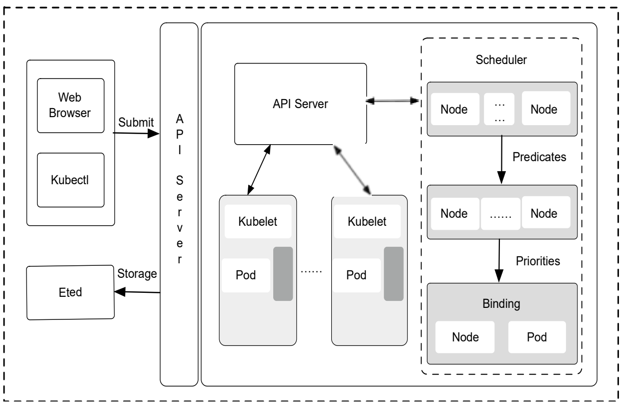


图 1　OpenShift 调度流程

Fig. 1　OpenShift scheduling process

1.2　OpenShift 调度算法

如图 1 所示，OpenShift 的 Kubernetes 容器调度引擎中 Scheduler 作为一个可插拔的调度模块[14]，用户可以很轻易地进行二次开发，扩充调度算法库。调度器相对于普通用户像一个黑盒，节点的资源状态和未调度应用需求作为输入，输出物理节点和应用的绑定策略。其调度算法中最重要的两个阶段分别是 Predicates(预选)和 Priorities(优选)。预选通过判断节点资源是否满足需求，筛选掉不合适的节点。优选则通过对剩余节点使用一系列的优先函数进行权重打分，选出评分最优的节点。其最终得分如式(1)所示。

$$s_i = (w_1 \times f_1) + (w_2 \times f_2) + \cdots + (w_p \times f_p) \tag{1}$$

其中 s_i 是预选后剩余节点 Node 的第 i 个节点最终得分，w_j 是给评分函数 f_j 的权重，f_j 是第 j 个评分函数。目前支持的优先级评分函数主要有以下几种:

a) LeastRequestedPriority 函数。节点空闲资源与节点资源总量的比值，即((节点资源总量-节点上已使用资源量-新应用资源需求)/总容量)来决定。CPU 和内存具有相同权重值，空闲资源越多，评分越高，取 CPU 和内存评分的均值。

b) BalancedResourceAllocation 函数。该平衡函数必须和 LeastRequestedPriority 评分函数一起使用。CPU 和内存使用率越接近的节点评分越高，用于简单的调节在部署 Pod 应用后各节点的 CPU 和内存资源利用率的均衡性。

c) SelectorSpreadPriority 函数。为了更好的应对灾难和宕机，对于属于同一个 Service、replication controller 的 Pod 副本，尽量调度到不同的节点上。

此外，调度函数还包括 NodePreferAvoidPodsPriority、NodeAffinityPriority、TaintTolerationPriority、EqualPriority、ImageLocalityPriority、MostRequestedPriority 等一系列特殊的评分函数。OpenShift 官方文档详细解释目前支持的评分函数和各函数的评分计算方法。

2　改进资源优先调度策略

2.1　改进后的优先调度流程

OpenShift 容器云平台默认的调度算法过于简单，不能满足生产环境中用户多样化的调度需求。其优选阶段仅考虑 CPU 和内存的空闲率会造成物理机的其他维度资源过度消耗，如磁盘、带宽等，物理节点将不能部署更多的应用，从而使数据中心产生巨大的资源浪费。然而，数据中心的应用类型可以根据应用请求资源的类型和比重进行识别，已出现很多云应用识别模型，主要包括 CPU、内存、磁盘和网络密集型等应用[15,16]。此外、默认调度器采用静态资源分配方式，应用在创建时申请好资源，不能随应用的实际需求进行调整。改进后优选阶段使用全新的评价函数体系对各节点资源空闲率进行评分计算，选取最合适的节点进行应用和节点的绑定，改进调度流程如图 2 所示。

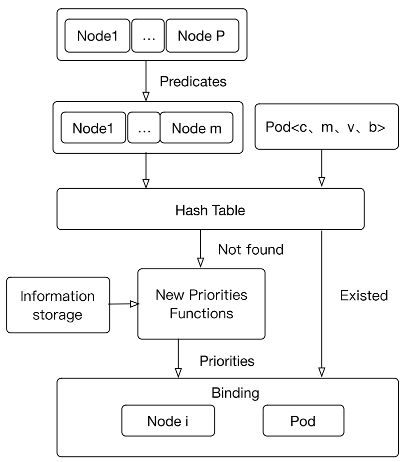


图 2　改进后 Priorities 部分调度流程

Fig. 2　Improved Priorities part scheduling process

预选阶段从集群 P 个节点中选出满足资源需求的 M 个节点。将应用的需求和满足的节点作为输入，查询 hash 表中该应用是否以前被创建过，如果在满足资源的节点上被调度过，直接绑定该节点和应用，调度结束。这对于重复的应用调度可以节省调度计算时间和缩短应用的启动时间，部署过的应用镜像文件避免重复下载。否则，进入新的函数评价模型。

信息存储模块存储所有物理节点当前已部署的 Pod 数量、应用的磁盘和带宽需求。未调度的 Pod 和各节点的资源情况可以通过 API Server 获取，输入到新的函数评价体系中，综合评价函数对各节点进行最终评分，评分最高的节点就是最合适的节点，应用绑定到节点。

2.2 全新函数评价模型

改进 OpenShift 资源调度的关键在于改进优选函数评价方式。由于 OpenShift 默认调度只考虑 CPU、内存的空闲率，本文根据应用的特性加入磁盘、网络带宽和已部署 Pod 应用的数量作为评分的依据，给预选出来的节点进行综合评分。提出了基于多维资源空闲率权重的调度方案 MRWS (multidimensional resource weight scheduling)。最终的评分函数体系可以分为两部分：多维资源空闲率评分模块和反馈模块。资源空闲模块表示该节点空闲资源数量，资源空闲率越高评分越高。反馈模块表示该节点多维资源利用率的均衡程度，资源之间利用越均衡评分越高。最终的得分定义为两者之和，在描述新的函数评价模型之前，先给出一些模型基本符号的定义和公式解释。

a) 定义多维资源的符号表示。假设经过预选后满足应用资源需求的物理节点数量有 m 台，用 $N=(n_1,n_2,n_3\ldots n_m)$ 表示。单个物理节点有 D 种资源，在本模型中，共有 CPU、内存、磁盘和网络带宽四种资源，因此 D=4。集群中物理节点总的资源表示为 $S=(s_1^d,s_2^d,s_3^d\ldots s_m^d), d\in D$，其中 s_i^d 表示节点 i 拥有的第 d 种资源类型总量。物理节点的资源使用量同样可以用上述方式表示为 $R=(r_1^d,r_2^d,r_3^d\ldots r_m^d), d\in D$，其中 r_i^d 表示节点 i 第 d 种资源的使用量。当前调度应用的资源需求可以定义为 p^d，表示容器应用对 d 种资源的需求。

b) 已部署 Pod 数量空闲率计算。由于已部署 pod 数量这个因素无法在 Pod 应用中加以限制和度量，也不确定物理节点能够部署应用的数量，因此本文按照如下的方式定义节点 Pod 的负载情况。应用数量之和如式(2)所示。

$$C=(\sum_{i=1}^{m}p_i+1) \tag{2}$$

其中：p_i 表示节点 i 上已经部署的 Pod 的数量，则节点 i 上 pod 部署空闲率 c_i 定义为如式(3)所示。

$$c_i=(1-\frac{p_i+1}{C}) \tag{3}$$

c) 多维资源权重系数。在实际系统中，物理节点上各种资源重要性往往具有较大差异，CPU 和内存是相对比较稀缺的资源，重要程度要高。定义权重系数 $\alpha_i, i\in[1,5]$、分别表示应用 CPU、内存、磁盘、网络带宽和已部署 Pod 数量负载的权重，且 $\alpha_1+\alpha_2+\alpha_3+\alpha_4+\alpha_5=1$。该权重系数根据不同的应用类型使用 FAHP 自动建模和求解，用户可以更加灵活的根据资源重要程度来调整资源调度模型。

d) 资源空闲模块的评分函数。资源空闲率模块表示物理节点资源空闲的量，空闲资源越多，模块评分越高。其最终的评分函数 v_i 如式(4)(5)所示。

$$v_i=\sum_{d=1}^{D}\alpha_d*k_i^d+\alpha_5*c_i \tag{4}$$

$$k_i^d=\frac{s_i^d-r_i^d-p^d}{s_i^d} \tag{5}$$

其中：k_i^d 表示节点 i 部署 p^d 后资源第 d 维资源空闲率。

e) 反馈模块评分函数。反馈模块表示多维资源的使用均衡程度，各维资源使用越均衡，反馈模块评分越高。其最终评分函数 b_i 如式(6)~(8)所示。

$$b_i=(\sum_{d=1}^{D}\alpha_d\frac{k_i^d}{k_v^d}+\alpha_5\frac{c_i}{c_v})/(D+1) \tag{6}$$

$$k_v^d=(\sum_{i=1}^{m}k_i^d)/m \tag{7}$$

$$c_v=(\sum_{i=1}^{m}c_i)/m \tag{8}$$

其中: b_i 表示反馈模块的最终评分。k_v^d 表示所有符合资源需求的 m 个物理节点上第 d 维种资源空闲率的均值，c_v 表示 pod 资源空闲率的均值。因此，k_i^d/k_v^d 就可以用来衡量 i 节点第 d 维资源空闲率在整个集群中不均衡性。比值越大，表示该节点第 d 种资源空闲就越多，资源利用越不均衡，从而反馈评分就越大。同样 c_i/c_c 表示 Pod 因素的不均衡性，最终单个物理节点的评分表示为 $f_i=v_i+b_i$。

f) 定义衡量节点不均衡度的表示。为了度量集群中 i 节点多维资源空闲率的负载均衡，使用方差作为衡量依据，如式(9)(10)所示。

$$u_i=\sqrt{\frac{1}{D+1}(\sum_{d=1}^{D})(k_i^d-k_v^d)^2+(c_i-c_v)^2}, \tag{9}$$

$$u_v=(\sum_{i=1}^{m}u_i)/m \tag{10}$$

其中：资源空闲率的方差 u_i 表示 i 节点资源空闲率的负载均衡，u_i 值越小，表示该节点各种资源的空闲率越均衡，u_v 表示整个集群的负载均衡。

3 FAHP 多维资源自动建模和参数求解

本节首先详细介绍 FAHP 建模流程，进而使用 FAHP 对影响容器调度的 CPU、内存、磁盘、网络带宽和已部署的 Pod 数量五个因素进行层次建模。然后自动构建容器应用的模糊成对比矩阵，根据资源需求比例调节模糊数使其判断矩阵满足一致性检验要求，判断矩阵最大特征值对应的特征向量归一化即为资源权重参数。求解所得的资源权重参数正是上一节中介绍容器应用的资源权重参数 $\alpha_i, i\in[1,5]$，使用这些参数作为调度优选阶段的容器应用评分函数多维资源的权重，最终实现容器云集群多维资源利用率的均衡，提升容器云平台综合资源利用率。

3.1 FAHP 建模流程

层次分析法[17, 18] (analytic hierarchy process，AHP)是美国运筹学家 Saaty 提出的一种简便、灵活而又实用的多准则权重决策分析方法。模糊层次分析法 FAHP 是 AHP 在构建

判断矩阵时考虑到人为判断的模糊性，在进行专家咨询时，往往只给出一个模糊量，比如三值判断、二值区间等。FAHP 模糊数集的隶属函数有正态分布、梯形函数、三角函数等。

本文采用三角函数，模糊数集 $\tilde{M}=(a,b,c)\, a \le b \le c$， $\tilde{M}$ 度量函数 $u_{\tilde{M}:}\, R->[0,1]$，其三角函数如式(11)所示。

$$u_{\tilde{M}}=\begin{cases} 0 & x<a \\ \dfrac{x-a}{b-a} & a \le x<b \\ \dfrac{c-x}{c-b} & b \le x \le x \\ 0 & x>c \end{cases} \tag{11}$$

文献[19,20]详细介绍了该算法的建模和求解方法，其主要的算法步骤如下:

a) 构建模糊成对比矩阵 $\tilde{A}$。构建一个成对比矩阵，使用模糊数 $\tilde{1} \sim \tilde{9}$ 表示两个因素的相对重要程度，值越大表示该因素相较于另一个因素越重要。 $\tilde{A}$ 元素如式(12)所示。

$$\tilde{a}_{ij}=\begin{cases} 1 & i=j \\ \tilde{1},\tilde{3},\tilde{5},\tilde{7},\tilde{9}\ or\ \tilde{1}^{-1},\tilde{3}^{-1},\tilde{5}^{-1},\tilde{7}^{-1},\tilde{9}^{-1} & i \ne i \end{cases} \tag{12}$$

b) 求解判断矩阵的最大特征值和特征向量。根据模糊成对比矩阵构建对应判断矩阵，并使用区间法求解判断矩阵，

$\forall \alpha \in [0,1]$， $\tilde{M}_{\alpha}=[l_{\alpha},u_{\alpha}]=[(b-a)\alpha+a,-(c-b)\alpha+c]$ 表示其割集，

成对比判断矩阵中元素的相对重要程度、模糊数以及 α 割集关系如表 1 所示。

表 1 模糊数和相对重要程度以及 α 割集关系

Table 1 Fuzzy numbers and corresponding importance and α cut set

模糊数	$\tilde{1}$	$\tilde{3}$	$\tilde{5}$	$\tilde{7}$	$\tilde{9}$
重要程度	同等重要	稍微重要	重要	比较重要	非常重要
$\tilde{M}$	(1,1,3)	(1,3,5)	(3,5,7)	(5,7,9)	(7,9,11)
l_{α}	1	1+2 α	3+2 α	5+2 α	7+2 α
u_{α}	3-2 α	5-2 α	7-2 α	9-2 α	11-2 α

给定优化参数 μ 表示求解判断矩阵的优化程度，可以根据模糊成对比矩阵获得对应的判断矩阵 *A*，判断矩阵 *A* 中的元素 a_{ij} 表示为 $\forall u \in [0,1],\, a_{ij}=(1-\mu)l_{\alpha}+\mu u_{\alpha}$,如式(13)所示。

$$a_{ij}=\begin{cases} 1 & \tilde{a}_{ij}=1 \\ (1-\mu)l_{\alpha}+\mu u_{\alpha} & \tilde{a}_{ij}\in\{\tilde{1},\tilde{3},\tilde{5},\tilde{7},\tilde{9}\} \\ (1-\mu)/u_{\alpha}+\mu/l_{\alpha} & \tilde{a}_{ij}\in\{\tilde{1}^{-1},\tilde{3}^{-1},\tilde{5}^{-1},\tilde{7}^{-1},\tilde{9}^{-1}\} \end{cases} \tag{13}$$

求解判断矩阵 *A* 的特征值和特征向量。

c) 一致性检验和归一化权重值。获取最大特征值后需要进行层次单排序和层次总排序的一致性检验，一致性指标和一致性比率公式如式(14)所示。

$$CI=\frac{\lambda_{\max}-n}{n-1} \qquad CR=\frac{CI}{RI} \tag{14}$$

其中： $\lambda_{\max}$ 是最大的特征值，*RI* 是随机一致性指标，随机一致性指标 *RI* 的值如表 2 所示。当 *CR*<0.1 时，构建的判断矩阵满足条件，否则需要调整模糊数，最大特征值对应的特征向量归一化就是所需的权重。

表 2 随机一致性指标 RI 值

Table 2 Random consistency indicator RI

n	RI	n	RI
1	0	4	0.90
2	0	5	1.12
3	0.58	6	1.24

3.2 FAHP 多维资源权重参数

新的函数评价模型中，将容器应用需求的 CPU、内存、磁盘、网络带宽和已部署 Pod 数作为影响应用调度的因素，预选出来的物理节点作为部署方案的候选节点。构造出如图 3 所示的层次结构模型，各个因素直接影响方案层的选择，是一个单层的模糊层次分析模型。目标层是需要选择一个合适的物理主机绑定应用，准则层是 CPU、内存、磁盘、网络带宽和部署应用 Pod 数，方案层是预选出来满足资源需求的物理主机。每个因素都对方案层施加影响，是一个全层次模型结构。通过这个模型，给不同的应用类型自动构建模糊成对比判断矩阵，生成对应的判断矩阵，并根据判断矩阵自动求解其多维资源的权重参数。

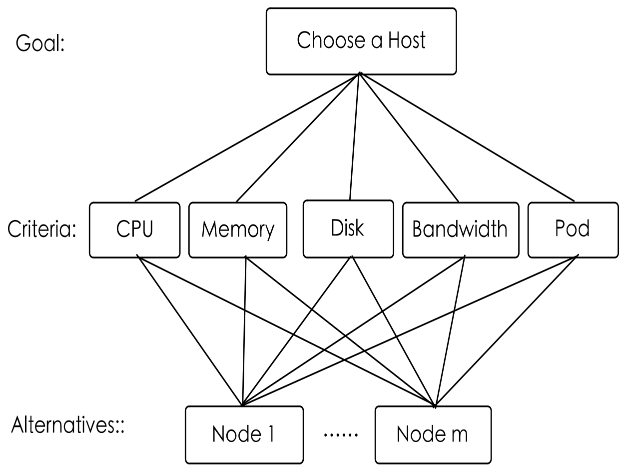


图 3 多维资源模糊层次结构

Fig. 3 Multidimensional resource FAHP

下面根据容器应用请求各维资源占节点资源的比例来作为该因素重要性的依据，请求资源所占的比例越大，表明该资源重要性越高，对应的模糊数越大，实验中单台物理机的资源配置如表 3 所示。

表 3 单台物理机配置

Table 3 Single physical machine configuration

CPU(M)	MEM(M)	Disk(G)	BW(M)
2400	16000	1024	1000

举例说明其自动求解过程，假设某个容器应用对资源的需求和所占节点资源的比重如表 4 所示。节点已部署 pod 的数量未知，该因素相对其他资源的重要性要低，可以用相对最小的模糊数表示该因素重要程度。

表 4 某容器应用资源需求

Table 4 Container application resource configuration

类型	CPU(M)	MEM(M)	Disk(G)	BW(M)	Pod
资源量	540	400	78	250	1
占比	0.225	0.025	0.075	0.4	-

根据容器应用资源需求的占比自动构建模糊成对比矩阵如式(15)所示，行和列分别是 CPU、内存、磁盘、带宽和 pod，从而构建一个 5*5 的方阵，根据式(13)获得对应的判断矩阵 *A*，求解判断矩阵最大特征值和特征向量。

$$\tilde{A}=\begin{bmatrix} 1 & \tilde{5} & \tilde{3} & \tilde{1} & \tilde{7} \\ \tilde{5}^{-1} & 1 & \tilde{1} & \tilde{5}^{-1} & \tilde{3} \\ \tilde{3}^{-1} & \tilde{1}^{-1} & 1 & \tilde{3}^{-1} & \tilde{5} \\ \tilde{1}^{-1} & \tilde{5} & \tilde{3} & 1 & \tilde{9} \\ \tilde{7}^{-1} & \tilde{3}^{-1} & \tilde{1} & \tilde{9}^{-1} & 1 \end{bmatrix} \tag{15}$$

判断矩阵 A 满足一致性检验的最大特征值对应的特征向量归一化的值就是所求的多维资源的权重值。如果所求的特征值不满足一致性检验，需要微调模糊数集，重新构建模糊成对比矩阵和对应的判断矩阵。实验中取 $\alpha=0.5$ 以及优化参数 $\mu=0.5$，构建对应的判断矩阵。最终求解的容器应用 CPU、内存、磁盘、带宽和已部署 pod 权重参数为(0.36,0.095,0.133,0.374,0.038)。任意的容器应用根据其资源需求可以通过该方法进行多维资源的参数求解，可以构建大量的测试应用。

4　实验分析

本文的实验在 ContainerCloudSim[21]上模拟容器云计算环境，在该平台上对调度策略进行仿真实验，并在实验室小规模集群上进行部署。实验从容器云集群资源利用率均衡度和容器云集群综合资源利用率两个方面对比 OpenShift 平台默认调度，随机 Random 调度以及本文提出的 MRWS 三种调度方式的优劣。单台物理机的配置如表 3 所示，容器应用使用 FAHP 对影响调度的 CPU、内存、磁盘、网络带宽和已部署 pod 五个因素进行建模。根据容器应用多维资源需求比例自动构建模糊成对比矩阵和判断矩阵，通过微调模糊数最终实现自动求解判断矩阵满足一致性检验的多维资源权重参数。调度中设置各资源的阈值为 0.95，一旦某个维度的资源超过该阈值，该节点将不能部署任何应用。首先构建 N 个用于调度容器应用如表 5 所示，其资源需求模拟容器云计算中 CPU、内存、磁盘和网络带宽四种密集型应用。

表 5　N 个容器应用资源需求

Table 5　N container application resource requirements

应用编号	CPU(M)	MEM(M)	Disk(G)	BW(M)	Pod
1	540	600	50	60	1
2	150	3600	15	40	1
3	120	800	220	20	1
..	..	..	..	..	..
N	60	400	20	240	1

使用上一节中介绍的自动构建模糊成对比矩阵和判断矩阵的方法构建满足一致性检验的判断矩阵，求解判断矩阵的最大特征值和对应的特征向量，特征向量归一化值即为资源的权重值。最终求解的表 5 中的 N 个容器应用对应的多维资源权重参数入表 6 所示。

表 6　N 个容器应用资源权重值

Table 6　N container application resource weight value

应用编号	CPU(M)	MEM(M)	Disk(G)	BW(M)	Pod
1	0.624	0.101	0.114	0.127	0.035
2	0.149	0.633	0.076	0.109	0.033
3	0.130	0.110	0.579	0.144	0.036
..	..	..	..	..	..
N	0.111	0.099	0.087	0.671	0.042

实验一设计了在相同数量节点不同应用数量的情况下，比较三种调度方式中容器云集群负载均衡度，其衡量指标如式(9)所示，实验结果如图 4 所示。

对图 4 的结果进行分析发现，相比于 OpenShift 默认调度方式和 Random 调度方式，本文提出的 MRWS 调度方式能够让容器云集群资源利用率更加均衡。这是因为 OpenShift 调度方式仅考虑 CPU 和内存资源利用的平衡性，没有考虑磁盘、网络带宽以及已部署的 pod 应用因素。一旦出现某个节点磁盘和网络带宽过载，将不能部署更多的应用，节点某个维度资源空闲较多，而其他维度资源过载。Random 调度方式集群资源利用均衡性最差，该方式完全没有考虑资源利用率的问题，容易造成单个节点过载，从而造成集群资源的浪费，对集群性能产生较大影响，降低了集群效率。

实验二设计了任务数量不同，比较三种调度方式容器云集群综合资源利用率，通过集群中 CPU、内存、磁盘和带宽利用率的均值来度量，实验结果如图 5 所示。

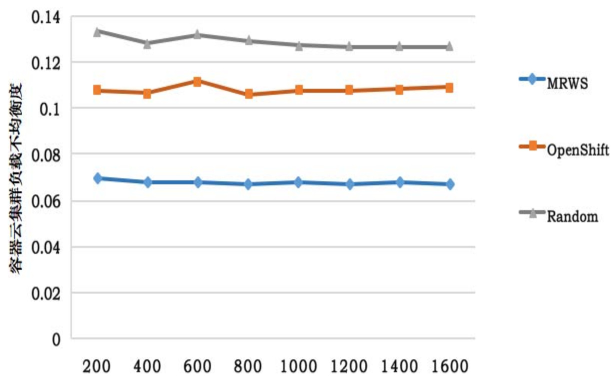


图 4　容器云集群资源负载不均衡度

Fig. 4　Container cloud cluster resource load imbalance degree

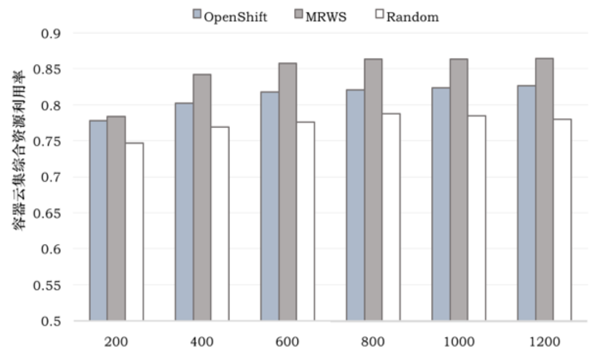


图 5　容器云集群综合资源利用率

Fig. 5　Container cloud cluster comprehensive resource utilization

对图 5 的结构进行分析，在集群规模和容器应用数量较小时，MRWS 调度方式优势并不明显，出现过载现象较少，集群整体的资源利用率都较低。随着应用数量的不断增大，Random 调度方式的劣势开始显现，随机部署会造成节点某一维度的资源过载，从而不能部署更多的应用，造成该维度资源的浪费。OpenShift 因为考虑了 CPU 和内存均衡性，效果比 Random 稍好，但是这种仅考虑 CPU 和内存的平衡性也会造成磁盘和网络带宽过载。在容器云中同时还会产生另一个问题，某个节点部署过多较小资源需求的应用，而另一个节点只部署了几个较大资源的应用，虽然两个节点整体的资源利用率较为均衡，但过多的容器应用会造成容器管理问题，从而降低集群性能，因此容器应用的部署也应该更加分散和均衡，这也是考虑已部署 pod 这个因素的原因。当容器应用数量达到一定数量后，MRWS 调度方式容器云集群资源综合利用率明显提升，所需的物理节点更少，从而实现提升资源利用率和集群性能。

5 结束语

通过对 OpenShift 私有容器云平台自带调度方式的研究，针对这种共享资源状态调度方式只考虑 CPU 和内存利用率的缺点，提出了一种改进的基于多维资源权重调度方法 MRWS 调度。该方法综合考虑容器应用 CPU、内存、磁盘、网络带宽和 Pod 数量，使用 FAHP 模糊层次分析法对影响调度的因素进行数学建模，根据容器应用对多维资源的需求所占比例自动构建模糊成对比矩阵和判断矩阵，自动求解满足一致性检验的多维资源权重参数。然后定义了多维资源的数学表示和物理节点权重的评分函数以及物理节点和集群资源利用率均衡度的衡量指标。最后设计实验对比 MRWS 调度方式和 OpenShift 默认调度方案以及 Random 调度的性能。实验表明新的调度方案能够有效提升单台物理机多维资源的负载均衡以及整个集群的负载均衡，从而提升整个集群的性能和资源利用率。在下一步的研究中，结合任务的迁移算法，需要考虑容器云上容器应用对资源的需求、网络 QoS、应用之间聚集性和互斥性、数据本地化、任务完成时间等更多因素。建立以应用类型为导向调度方式，进一步提升集群的性能和资源利用率。

参考文献：

[1] 浙江大学 SEL 实验室. Docker 容器与容器云 [M]. 2 版. 北京: 人民邮电出版社, 2016. （Software Engineering Lab of Zhejiang University.Docker and Kubernetes under the hood[M].2nd ed.Beijing:Posts & Telecom Press,2016.）

[2] Seo K T, Hwang H S, Moon I Y, *et al.* Performance comparison analysis of linux container and virtual machine for building cloud [J]. Advanced Science and Technology Letters, 2014, 66 (2): 105-111.

[3] Casalicchio E. Autonomic orchestration of containers: Problem definition and research challenges [C]// Proc of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools. Guimaraes: ICST Press, 2017: 287-290

[4] 林伟伟, 齐德昱. 云计算资源调度研究综述 [J]. 计算机科学, 2012, 39 (10): 1-6. (Lin Weiwei, Qi Deyu. Survey of resource scheduling in cloud computing [J]. Computer Science, 2012, 39 (10): 1-6)

[5] Quang-Hung N, Nien P D, Nam N H, *et al.* A genetic algorithm for power-aware virtual machine allocation in private cloud [C]// Proc of Information and Communication Technology-EurAsia Conference. Berlin: Springer, 2013: 183-191.

[6] Soni G, Kalra M. A novel approach for load balancing in cloud data center [C]// Proc of Advance Computing Conference. Piscataway, NJ: IEEE Press, 2014: 807-812.

[7] Arianyan E, Taheri H, Sharifian S. Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers [J]. Computers & Electrical Engineering, 2015, 47 (10): 222-240.

[8] You X, Xu X, Wan J, *et al.* RAS-M: resource allocation strategy based on market mechanism in cloud computing [C]// Proc of ChinaGrid Annual Conference. Piscataway, NJ: IEEE Press, 2009: 256-263.

[9] Jansen C, Witt M, Krefting D. Employing docker swarm on openstack for biomedical analysis [C]// Proc of International Conference on Computational Science and Its Applications. Switzerland: ICCSA Press, 2016: 303-318.

[10] Hindman B, Konwinski A, Zaharia M, *et al.* Mesos: A platform for fine-grained resource sharing in the data center [C]// Proc of the 8th USENIX conference on Networked systems design and implementation. Berkeley: USENIX Association Press, 2011: 429-483.

[11] Ghodsi A, Zaharia M, Hindman B, *et al.* Dominant resource fairness: fair allocation of multiple resource types [C]// Proc of USENIX Conference on Networked Systems Design and Implementation. Berkeley: USENIX Association Press, 2012: 323-336.

[12] Verma A, Pedrosa L, Korupolu M, *et al.* Large-scale cluster management at Google with Borg [C]// Proc of the 10th European Conference on Computer Systems. New York: ACM Press, 2015: 18-35.

[13] Lossent A, Peon A R, Wagner A. PaaS for Web applications with OpenShift Origin [C]// Proc of Journal of Physics: Conference Series. UK: IOP Publishing, 2017: 37-45.

[14] Rensin D K. Kubernetes-scheduling the Future at Cloud Scale [M]. CA: O'Reilly and Associates. 2015.

[15] 麻双克, 周兰凤. 云计算环境下基于优先级的 IO 和网络密集型应用调度策略 [J]. 上海理工大学学报, 2017, 39 (5): 505-510. (Ma Shuangke, Zhou Lanfeng. Scheduling strategy of IO and network intensive applications based on the priority in cloud environment[J]. Journal of Shanghai University of Technology, 2017, 39 (5): 505-510.)

[16] Peng J, Dai Y, Rao Y, *et al.* Research on processing strategy for CPU-intensive application [J]. Journal of Systems Architecture, 2016, 70 (10): 39-47.

[17] Saaty T L, Decision H T M A. The analytic hierarchy process [J]. European Journal of Operational Research, 1990, 48 (2): 9-26.

[18] 邓雪, 李家铭, 曾浩健, 等. 层次分析法权重计算方法分析及其应用研究 [J]. 数学的实践与认识, 2012, 42 (7): 17-24. (Deng Xue, Li Jiaming, Zeng Haojian *et al.* Analysis and application of weight calculation method of analytic hierarchy process [J]. Mathematics in Practice and Theory, 2012, 42 (7): 17-24)

[19] Kwong C K, Bai H. A fuzzy AHP approach to the determination of importance weights of customer requirements in quality function deployment [J]. Journal of Intelligent Manufacturing, 2002, 13 (5): 367-377.

[20] Peng H, Yang G X, Cai L, *et al.* Cloud test environment deployment based on the needs of individual users [C]// Information Science and Service Science and Data Mining. Piscataway, NJ: IEEE Press, 2013: 562-567.

[21] Piraghaj S F, Dastjerdi A V, Calheiros R N, *et al.* ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers [J]. Software: Practice and Experience, 2017, 47 (4): 505-521